Nicht für den Prüfling bestimmt

I Erläuterungen

Voraussetzungen gemäß KCBG und Abiturerlassen BG jeweils in der für den Abiturjahrgang geltenden Fassung

Standardbezug
Die nachfolgend ausgewiesenen Kompetenzbereiche sind für die Bearbeitung der jeweiligen Aufgabe besonders bedeutsam. Darüber hinaus können weitere, hier nicht ausgewiesene Kompetenzbereiche für die Bearbeitung der Aufgabe nachrangig bedeutsam sein, zumal die Kompetenzbereiche in engem Bezug zueinanderstehen. Die Operationalisierung des Bezugs zu den Kompetenzbereichen des Standardbezugs erfolgt in Abschnitt II.

Aufgabe	Kompetenzbereiche				
	K1	K2	K3	K4	K5
1.1		X			
1.2		X		X	
1.3		X	X		
1.4.1		X			X
1.4.2		X	X		
1.4.3		X		X	
1.5.1		X	X		
1.5.2		X	X	X	
1.6		X		X	
2.1.1	X	X			X
2.1.2		X	X		
2.1.3	X		X		
2.2.1			X	X	
2.2.2				X	
2.2.3			X	X	
2.2.4			X	X	
2.2.5				X	X
2.3		X	X		

Inhaltlicher Bezug
Die nachfolgend ausgewiesenen Themenfelder sind die wesentliche inhaltliche Grundlage für die vorliegenden Aufgaben. Darüber hinaus können weitere, hier nicht explizit ausgewiesene Themenfelder für die Bearbeitung nachrangig bedeutsam sein.
Q1: Objektorientierte Softwareentwicklung
Q2: Datenbanksysteme
Q3: Datenkommunikation
verbindliche Themenfelder: Objektorientierte Modellierung (Q1.1), Implementierung von Klassen und Assoziationen (Q1.2), Konzeptionelle und logische Modellierung einer Datenbank (Q2.1), Datenabfrage und Datenmanipulation mit SQL (Q2.2), Kommunikation in Rechnernetzen (Q3.2)

II Lösungshinweise

In den nachfolgenden Lösungshinweisen sind alle wesentlichen Gesichtspunkte, die bei der Bearbeitung der einzelnen Aufgaben zu berücksichtigen sind, konkret genannt und diejenigen Lösungswege aufgezeigt, welche die Prüflinge erfahrungsgemäß einschlagen werden. Selbstverständlich sind jedoch Lösungswege, die von den vorgegebenen abweichen, aber als gleichwertig betrachtet werden können, ebenso zu akzeptieren.

Aufg.	erwartete Leistungen	BE		
		I	II	III
1.1	benennen, erläutern Beide Assoziationen sind Ganzes-Teile-Assoziationen. Zwischen Beitrag und Bild ist eine Aggregation modelliert (nicht ausgefüllte Raute). Wird ein Beitrag gelöscht, bleibt das Bild im System erhalten. Bei der Aggregation ist die Lebensdauer der Teile, hier der Bilder, nicht an die Lebensdauer des Ganzen, hier der Beitrag, gebunden. Zwischen Beitrag und Text ist eine Komposition modelliert (ausgefüllte Raute). Bei der Komposition ist die Lebensdauer der Teile, hier des optionalen Textes, an die Lebensdauer des Ganzen, des Beitrags gebunden. Wird der Beitrag gelöscht, wird auch der Text gelöscht.			
	benennen	1		
	erläutern		2	
1.2	überführen, implementieren			

```
public class Nutzer {
  private String benutzerName;
  private String passwort;
  private String emailAdresse;
  private DateTime zuletztAktiv;
  private List<Beitrag> beitraege;
  private List<Nutzer> abonnenten;
  private List<Nutzer> abonnierteNutzer;
  private List<Bild> bilder;

  public Nutzer(String name, String passwort,
         String email) {
    this.benutzerName = name;
    this.passwort = passwort;
    this.emailAdresse = email;
    this.zuletztAktiv = new DateTime();
    this.beitraege = new List<>();
    this.abonnenten = new List<>();
    this.abonnierteNutzer = new List<>();
    this.bilder = new List<>();
  }

  public Beitrag erstelleBeitrag(String titel, Bild bild) {
    this.zuletztAktiv = new DateTime();
    Beitrag beitrag = new Beitrag(this, titel, Bild);
    this.beitraege.add(beitrag);
    return beitrag;
  }

  public Beitrag erstelleBeitrag(
                    String titel, Bild bild, String text){
    return erstelleBeitrag(titel, bild).erstelleText(text);
  }
```

<table>
<tr><th rowspan="2">Aufg.</th><th rowspan="2">erwartete Leistungen</th><th colspan="3">BE</th></tr>
<tr><th>I</th><th>II</th><th>III</th></tr>
<tr><td></td><td>

```
  public void abonnieren(Nutzer n) {
    if (n != this){
      this.zuletztAktiv = new DateTime();
      if (!this.abonnierteNutzer.contains(n)) {
        this.abonnierteNutzer.add(n);
        n.abonnenten.add(this);
      }
    }
  }

  public void like(Beitrag beitrag) {
    if(beitrag.getAutor() != this)
      beitrag.like();
      this.zuletztAktiv = new DateTime();
  }
  public void hinzufuegenBild(Bild bild) {
    bilder.add(bild);
    zuletztAktiv = new Datetime();
  }
}

public class Beitrag {
  private DateTime gepostet;
  private String titel;
  private int anzahlLikes;
  private List<Bild> bilder;
  private Text text = null;
  private Nutzer autor;

  public Beitrag(Nutzer autor, String titel, Bild bild) {
    this.autor = autor;
    gepostet = new DateTime();
    this.titel = titel;
    bilder = new List<>();
    this.hinzufuegen(bild);
    anzahlLikes = 0;
  }

  public void hinzufuegen(Bild bild) {
    bilder.add(bild);
  }

  public void erstelleText(String text) {
    if(this.text == null)
      this.text = new Text(text);
  }

  public void like() {
    anzahlLikes++;
  }
}
```

überführen
implementieren</td><td>5</td><td>4</td><td>4</td></tr>
</table>

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE		
		I	II	III
1.3	entwickeln, zeichnen **ermittleAbonnierteNutzerMitNeuenBeitraegen(n : Nutzer)** leere Nutzer-Liste ergebnis anlegen für jeden Nutzer **abon** in abonnierteNutzer von n für jeden Beitrag **b** in beitraege von **abon** Beitrag **b** wurde gepostet nachdem Nutzer n zuletzt aktiv war ? J: Nutzer **abon** der Liste ergebnis hinzufügen break N: Ø Liste ergebnis zurückgeben entwickeln zeichnen	 2	2 1	2
1.4.1	beschreiben Die Socket-Klassen verwendet man, um Daten senden und empfangen zu können. Sie dienen dem Aufbau einer bidirektionalen Verbindung. Anschließend führen die erzeugten Socket-Objekte Methoden zum Schreiben und Lesen von Texten aus. `Server`: Die Klasse `Server` implementiert einen Server, der an einem Port horcht und Verbindungsanfragen von Clients über einen Server-Socket entgegennimmt. Wurde eine Verbindungsanfrage akzeptiert, wird ein Socket geliefert und ein Objekt der Klasse `ServerThread` für diesen Client erzeugt und gestartet. Der `Server` erhält bei der Erzeugung eine Referenz auf ein Objekt der Klasse `SocialMediaPlatform`, die die Anwendungsfälle der Social-Media-Plattform in Methoden implementiert. Diese Referenz wird an den `ServerThread` weitergegeben, der die Methoden aufruft. `ServerThread`: Die Klasse `ServerThread` erbt von der Klasse `Thread`, damit der Client-Server-Dialog in einem nebenläufigen Prozess ausgeführt wird. Die `run`-Methode von `Thread` wird deshalb überschrieben. Ein Thread wird stets durch den Aufruf der Methode `start()` erzeugt und initialisiert. Dabei wird schließlich die Methode `run()` aufgerufen. Bei der Anmeldung schreibt der Client die Anforderung auf seinen Socket (`write`-Methode), der `ServerThread` liest die Anforderung von seinem Socket mit `readline`() und ruft die `anmelden`-Methode der Klasse `SocialMediaPlatform` auf. Die Antwort wird über dieselbe Verbindung in umgekehrter Richtung geschickt.	2	2	

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.4.2	entwickeln, zeichnen			
	entwickeln		2	4
	zeichnen	2	2	

Nicht für den Prüfling bestimmt

<table>
<tr><th rowspan="2">Aufg.</th><th rowspan="2">erwartete Leistungen</th><th colspan="3">BE</th></tr>
<tr><th>I</th><th>II</th><th>III</th></tr>
<tr><td>1.4.3</td><td>überführen<pre>
public class Server {
  private ServerSocket serverSocket;
  private SocialMediaPlatform smp;

  public Server(int port, SocialMediaPlatform smp) {
    this.smp = smp;
    serverSocket = new ServerSocket(port);
  }

  public void runServer() {
    while (true){
      Socket clientSocket = serverSocket.accept();
      new ServerThread(clientSocket, smp).start();
    }
  }
}
</pre></td><td>2</td><td>3</td><td></td></tr>
<tr><td>1.5.1</td><td>entwickeln
Für das Passwort wird ein 12-stelliges char-Array deklariert. Die Indizes der freien Stellen werden in einer Liste vom Datentyp Integer verwaltet. Zu Beginn ist das Passwort-Array leer, deshalb werden alle Indizes (0 – 11) in die Liste eingetragen.
Die vier Sonderzeichen sind in einem Zeichen-Array gegeben. Großbuchstaben und Ziffern werden mittels einer Zufallszahl erzeugt, für die der geforderte Wertebereich festgelegt wird. Anschließend entstehen aus den gecasteten zufälligen <code>int</code>-Werten die geforderten Zeichen.
Beim Eintragen eines Zeichens in das Passwort wird zunächst ein Element der Liste per Zufall ermittelt. Dieses Element liefert den Index im Passwort und das Zeichen wird bei diesem Index eingetragen. Anschließend wird dieser verwendete Index aus der Liste gestrichen.
Am Ende werden Kleinbuchstaben auf alle noch freien Stellen geschrieben, deren Indizes noch in der Liste stehen.

Hinweis: Der hier entwickelte und beschriebene Algorithmus verwendet explizite und automatische Typecasts zwischen int und char.</td><td></td><td>2</td><td>3</td></tr>
<tr><td>1.5.2</td><td>Implementieren<pre>
public char[] generierePasswort() {
    char[] pwArray = new char[12];
     for(int i = 0; i < pwArray.length; i++) {
       pwArray[i] = (char)(nextInt(26) + 97);
    }
    int z1 = nextInt(12);
    int z2 = nextInt(12);
    while(z1 == z2) {
      z2 = nextInt(12);
    }
    int z3 = nextInt(12);
    while(z3 == z1 || z3 == z2) {
      z3 = nextInt(12);
    }
    pwArray[z1] = (char)(nextInt(26) + 65);
    pwArray[z2] = (char)(nextInt(10) + 48);
</pre></td><td></td><td>3</td><td>5</td></tr>
</table>

Aufg.	erwartete Leistungen	BE I	BE II	BE III

	``
	    pwArray[z3] = (char)(nextInt(4) + 35);
	    return pwArray;
	  }
	``
	Hinweis: Auch andere Algorithmen sind zulässig.			

1.6	implementieren
	``
	public int registrieren(String name, String passwort,
	                         String email) {
	  int returncode = 0;
	  if (this.sucheNutzer(name) == null) {
	    Nutzer n_mail = null;
	    for (Nutzer n: this.nutzer) {
	      if(n.getEmailAdresse().equals(email)){
	         n_mail = n;
	         break;
	       }
	    }
	    if (n_mail == null) {
	      nutzer.add(new Nutzer(name, passwort, email));
	    } else {
	      returncode = -2; // email-Adresse bereits benutzt
	    }
	  } else {
	      returncode = -1; // Nutzername bereits vergeben
	  }
	  return returncode;
	}
	``
		1	3	1

	Summe 60	**15**	**26**	**19**

Aufg.	erwartete Leistungen	BE I	BE II	BE III
2.1.1	angeben 1. Jeder Entitätstyp wird als Relation dargestellt. Jede Relation benötigt einen Primärschlüssel. 2. Jede n:m-Beziehung zwischen zwei Entitätstypen wird in eine eigene Tabelle transformiert, wobei beide Primärschlüssel der Entitätstypen als Fremdschlüssel in die Beziehungstabelle eingehen und zusammen den Primärschlüssel bilden. 3. Jede 1:n-Beziehung wird so transformiert, dass der Primärschlüssel des 1-Entitätstyps Fremdschlüssel des n-Entitätstyps wird. 4. Hat eine Beziehung zwischen zwei Entitätstypen ein oder mehrere Attribute, werden diese zu Attributen in der Relation, in der der Fremdschlüssel entsteht (auf der n-Seite einer 1:n-Beziehung, bzw. in der Beziehungstabelle einer n:m-Beziehung bzw. in der Relation, in die eine 1:1 Beziehung transformiert wird).	4		

2.1.2	überführen
	`Nutzer (benutzerName, passwort, email, zuletztAktiv)`
	`Beitrag (beitragid, erstelltAm, titel, text, autor#)`
	`NutzerLikesBeitrag (benutzerName#, beitragid#, zeitstempel)`
	`Bild (bildid, dateiname)`
	`BeitragEnthaeltBild (beitragid#, bildid#)`
	(Primärschlüssel unterstrichen: benutzerName, beitragid, benutzerName#, beitragid#, bildid, beitragid#, bildid#)
		3	2	

Aufg.	**erwartete Leistungen**	**BE** I	**BE** II	**BE** III
2.1.3	erklären Daten in einer Datenbank müssen vollständig, korrekt und widerspruchsfrei sein. Diese drei Eigenschaften gewährleisten die Integrität in einer relationalen Datenbank. Jede Tabelle muss einen Primärschlüssel haben. Wenn Beziehungen aus dem ERM in Relationen überführt werden, muss dieser als Fremdschlüssel in eine andere Tabelle aufgenommen werden und stellt so die Verknüpfung zwischen den Datensätzen der einen Tabelle mit denen der anderen Tabelle her. Die referentielle Integrität besagt, dass alle Attributwerte eines Fremdschlüssels auch als Attributwerte eines Primärschlüssels vorhanden sein müssen. Fremdschlüssel müssen immer auf existierende Datensätze verweisen. Beispielsweise muss eine `beitragid` in der Tabelle `BeitragEnthaeltBild` immer auf eine tatsächliche `beitragid` eines Beitrags in der Tabelle `Beitrag` verweisen. Die referentielle Integrität wäre verletzt, wenn in der Tabelle `Beitrag` unter dem Fremdschlüssel `autor` der Name „Müller98“ eingetragen, aber in der Tabelle `Nutzer` in der Spalte des Primärschlüssels `benutzerName` nicht zu finden ist.		2	2

2.2.1	formulieren
	`SELECT BE.titel, BEB.bildid, B.dateiname`
	`FROM   Beitrag BE JOIN BeitragEnthaeltBild BEB USING(beitragid)`
	`       JOIN Bild B USING(bildid)`
	`WHERE  BE.autor = 'Anna';`
			3	

2.2.2	angeben
	`INSERT INTO Nutzer`
	`VALUES ('Clark', 'ert4$alPhawe', 'Clark@familiy.com', NOW());`
	
	`INSERT INTO Beitrag`
	`VALUES (42, NOW(), 'Best of 2022', null, 'Clark');`
	
	`INSERT INTO BeitragEnthaeltBild (beitragid, bildid)`
	`VALUES (42,3);`
		4		

2.2.3	implementieren
	`SELECT B.titel AS Beitragstitel, B.autor AS 'Autor:in',`
	`       COUNT(N.beitragid) AS 'Anzahl Likes im Januar 2023'`
	`FROM   NutzerLikesBeitrag N JOIN Beitrag B USING(beitragid)`
	`WHERE  N.zeitstempel LIKE '2023-01%'`
	`GROUP BY N.beitragid`
	`HAVING COUNT(N.beitragid) > 10`
	`ORDER BY COUNT(N.beitragid) DESC;`
			3	2

2.2.4	entwickeln
	`SELECT bildid, dateiname`
	`FROM   Bild`
	`WHERE  bildid NOT IN`
	`(SELECT bildid FROM BeitragEnthaeltBild);`
			1	2

<table>
<tr><th rowspan="2">Aufg.</th><th rowspan="2">erwartete Leistungen</th><th colspan="3">BE</th></tr>
<tr><th>I</th><th>II</th><th>III</th></tr>
<tr><td>2.2.5</td><td>erörtern
Die angegebene SQL-Anweisung erzeugt nicht die gewünschte Ergebnistabelle, denn bei fehlender Verknüpfung über ein gemeinsames Feld wird von den Tabellen <code>Nutzer</code> und <code>Beitrag</code> das kartesische Produkt gebildet.
Die notwendige Verknüpfung realisiert man durch einen JOIN über die Primärschlüssel-Fremdschlüssel-Beziehung zwischen <code>benutzername</code> und <code>autor</code>. Weil darüber hinaus alle Nutzer in der Ergebnistabelle enthalten sein sollen, muss ein LEFT JOIN angewendet werden.

formulieren
<pre>SELECT benutzerName AS Benutzername, titel AS Beitragstitel,
       erstelltAm AS Erstellungsdatum
FROM   Nutzer N LEFT JOIN Beitrag B ON
       N.benutzerName = B.autor</pre></td><td></td><td>

2</td><td>

2</td></tr>
</table>

Aufg.	erwartete Leistungen	BE I	BE II	BE III
2.3	entwickeln, zeichnen 			
	entwickeln		3	2
	zeichnen	2	1	
	Summe 40	**13**	**17**	**10**

III Bewertung und Beurteilung

Die Bewertung und Beurteilung erfolgt unter Beachtung der nachfolgenden Vorgaben nach § 33 der Oberstufen- und Abiturverordnung (OAVO) in der jeweils geltenden Fassung. Bei der Bewertung und Beurteilung der sprachlichen Richtigkeit in der deutschen Sprache sind die Bestimmungen des § 9 Abs. 12 Satz 3 OAVO in Verbindung mit Anlage 9b anzuwenden.

Bei der Bewertung und Beurteilung der Übersetzungsleistung in den Fächern Latein und Altgriechisch sind die Bestimmungen des § 9 Abs. 14 OAVO in Verbindung mit Anlage 9c anzuwenden.

Der Fehlerindex ist nach Anlage 9b zu § 9 Abs. 12 OAVO zu berechnen. Für die Ermittlung der Punkte nach Anlage 9a zu § 9 Abs. 12 OAVO sowie Anlage 9c zu § 9 Abs. 14 OAVO wird jeweils der ganzzahlige nicht gerundete Prozentsatz bzw. Fehlerindex zugrunde gelegt.

Für die Bewertung in den modernen Fremdsprachen ist der „Erlass zur Bewertung und Beurteilung von schriftlichen Arbeiten in allen Grund- und Leistungskursen der neu beginnenden und fortgeführten modernen Fremdsprachen in der gymnasialen Oberstufe, dem beruflichen Gymnasium, dem Abendgymnasium und dem Hessenkolleg“ vom 7. August 2020 (ABl. S. 519) zugrunde zu legen. Demnach erfolgt die Bewertung und Beurteilung mit der Maßgabe, dass lediglich bei der Ermittlung des Prüfungsergebnisses (Note) aus Prüfungsteil 1 und 2 gerundet wird.

Darüber hinaus sind die Vorgaben der Erlasse „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen (Abiturerlass)“, „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen im beruflichen Gymnasium (fachrichtungs-/ schwerpunktbezogene Fächer) (Abiturerlass BG)“ und „Durchführungsbestimmungen zum Landesabitur“ in der für den Abiturjahrgang geltenden Fassung zu beachten.

Als Kriterien für die Bewertung und Beurteilung dienen unter Beachtung der Zielsetzung der gymnasialen Oberstufe nach § 1 Abs. 2 OAVO neben dem Inhaltlichen auch die in den Kerncurricula genannten überfachlichen Kompetenzen, insbesondere die Sprachkompetenz und Wissenschaftspropädeutik; dies zeigt sich u.a. in qualitativen Merkmalen wie Strukturierung, Differenziertheit, (fach-)sprachlicher Gestaltung und Schlüssigkeit der Argumentation.

Im Fach Praktische Informatik besteht die Prüfungsleistung aus der Bearbeitung eines Vorschlags, wofür insgesamt maximal 100 BE vergeben werden können. Ein Prüfungsergebnis von **5 Punkten (ausreichend)** setzt voraus, dass mindestens 45% der zu vergebenden BE erreicht werden. Ein Prüfungsergebnis von **11 Punkten (gut)** setzt voraus, dass mindestens 75% der zu vergebenden BE erreicht werden.

Gewichtung der Aufgaben und Zuordnung der Bewertungseinheiten zu den Anforderungsbereichen

Aufgabe	Bewertungseinheiten in den Anforderungsbereichen			Summe
	AFB I	AFB II	AFB III	
1	15	26	19	**60**
2	13	17	10	**40**
Summe	**28**	**43**	**29**	**100**

Die auf die Anforderungsbereiche verteilten Bewertungseinheiten innerhalb der Aufgaben sind als Richtwerte zu verstehen.